



Luigi

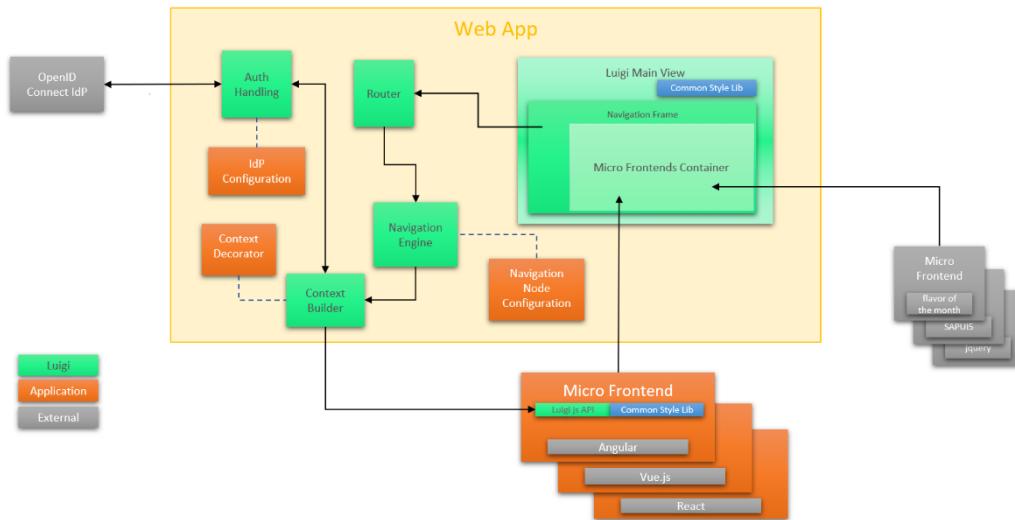
Es un framework de micro-frontends de código abierto escrito en Svelte. Permite crear una interfaz de usuario y navegación consistentes, al mismo tiempo que ofrece funciones adicionales para facilitar el desarrollo. Su apariencia se basa en los estilos de la biblioteca Fundamentals. Luigi es agnóstico a la tecnología, lo que permite construir la aplicación y agregar micro-frontends utilizando React, Angular, UI5 o cualquier otra tecnología.

[Luigi Fiddle](#) es un espacio de pruebas donde puede familiarizarse con Luigi.

SAP/Luigi: Microfrontend Framework



Este diagrama representa la arquitectura básica de Luigi:



Flujo de la aplicación:

1. **Autenticación:** Los usuarios se autentican a través de un proveedor de identidad.
2. **Enrutamiento:** El router dirige las solicitudes a los componentes apropiados.
3. **Configuración:** Se carga la configuración de la aplicación y del usuario.
4. **Construcción del contexto:** Se crea un contexto con información relevante para la solicitud.
5. **Motor de navegación:** Gestiona la navegación entre diferentes vistas de la aplicación.
6. **Configuración del nodo de navegación:** Define las opciones de navegación disponibles.
7. **Contenedor de Microfrontends:** Aloja y gestiona los microfrontends.
8. **Microfrontends:** Componentes independientes responsables de partes específicas de la interfaz de usuario.

Esta es una pequeña lista de términos de Luigi que serán útiles al hacer el taller:

Luigi Core - se refiere a la aplicación principal dentro de la cual se muestra un micro-frontend. Incluye la navegación superior y lateral y todas las demás configuraciones relacionadas con la aplicación principal.

Luigi Client - un término que abarca todas las configuraciones relacionadas con el micro-frontend ofrecidas por Luigi. Los micro-frontends son configurables a través de la API de Luigi Client.

Parámetros - los parámetros son los elementos utilizados para configurar tu aplicación Luigi. Lee la documentación de Luigi para obtener una lista completa de los parámetros de configuración de [navegación](#), [autorización](#) y [configuración general](#).

Nodos de navegación - los enlaces individuales de la navegación lateral en Luigi.

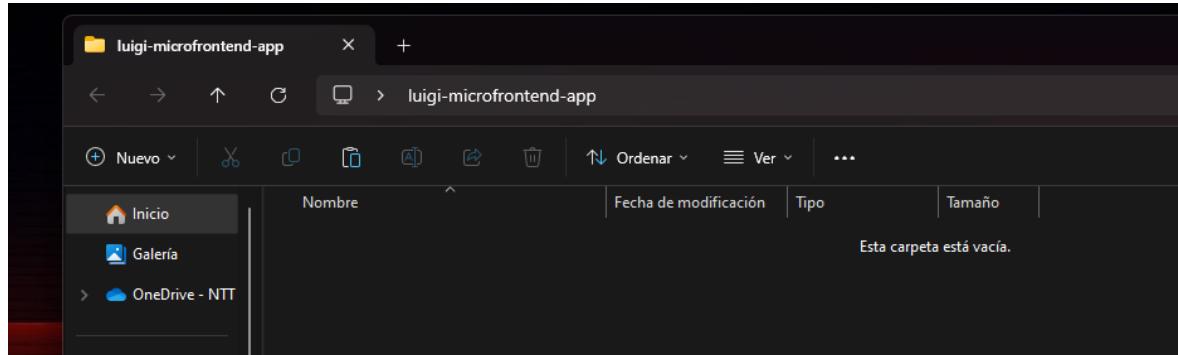
Contextos - los contextos son parámetros de Luigi que te permiten pasar objetos al micro-frontend.

Vistas - otro nombre para los micro-frontends.

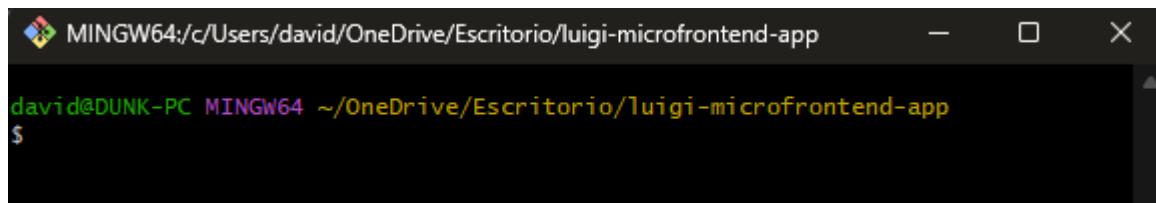
Taller práctico

Vamos a crear un proyecto base de microfrontends en Luigi

1. Creamos una carpeta en la ubicación de su preferencia, con el nombre: luigi-microfrontend-app

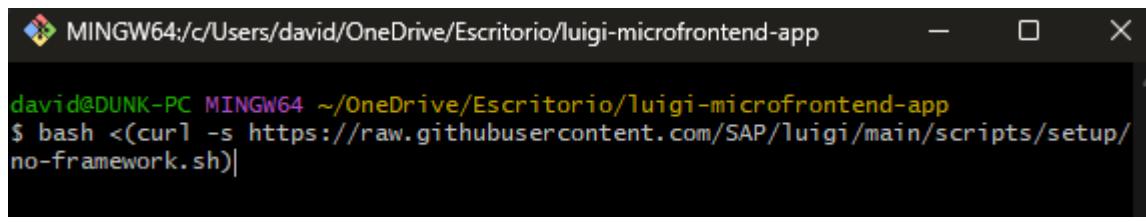


2. Abriremos la terminal bash en la ubicación de la carpeta que acabamos de crear, en este caso, usare la cmd de Windows.



3. Pegamos el siguiente comando:

```
bash <(curl -s https://raw.githubusercontent.com/SAP/luigi/main/scripts/setup/no-framework.sh)
```



Este comando nos pedirá darle un nombre a la carpeta que se va a crear, en este caso la llamaremos **luigi-core**

SAP/Luigi: Microfrontend Framework

```
MINGW64:/c/Users/david/OneDrive/Escritorio/luigi-microfrontend-app - X
david@dUNK-PC MINGW64 ~/OneDrive/Escritorio/luigi-microfrontend-app
$ bash <(curl -s https://raw.githubusercontent.com/SAP/luigi/main/scripts/setup/no-framework.sh)

Installing Luigi with static files and basic configuration

Luigi project folder name: luigi-core
```

Damos enter, y esperamos a que termine su instalación.

```
MINGW64:/c/Users/david/OneDrive/Escritorio/luigi-microfrontend-app - X
          Dload Upload Total Spent Left Speed
100 7406 100 7406 0 0 31123 0 ---:---:---:---:---:--- 31248
  % Total  % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 772 100 772 0 0 3478 0 ---:---:---:---:---:--- 3477
  % Total  % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 760 100 760 0 0 3183 0 ---:---:---:---:---:--- 3206
  % Total  % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 1340 100 1340 0 0 6072 0 ---:---:---:---:---:--- 6118
  % Total  % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 708 100 708 0 0 3173 0 ---:---:---:---:---:--- 3174
  % Total  % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 410 100 410 0 0 1916 0 ---:---:---:---:---:--- 1924
Running server with command: npm run start

> luigi-example-js@0.1.0 start
> serve public

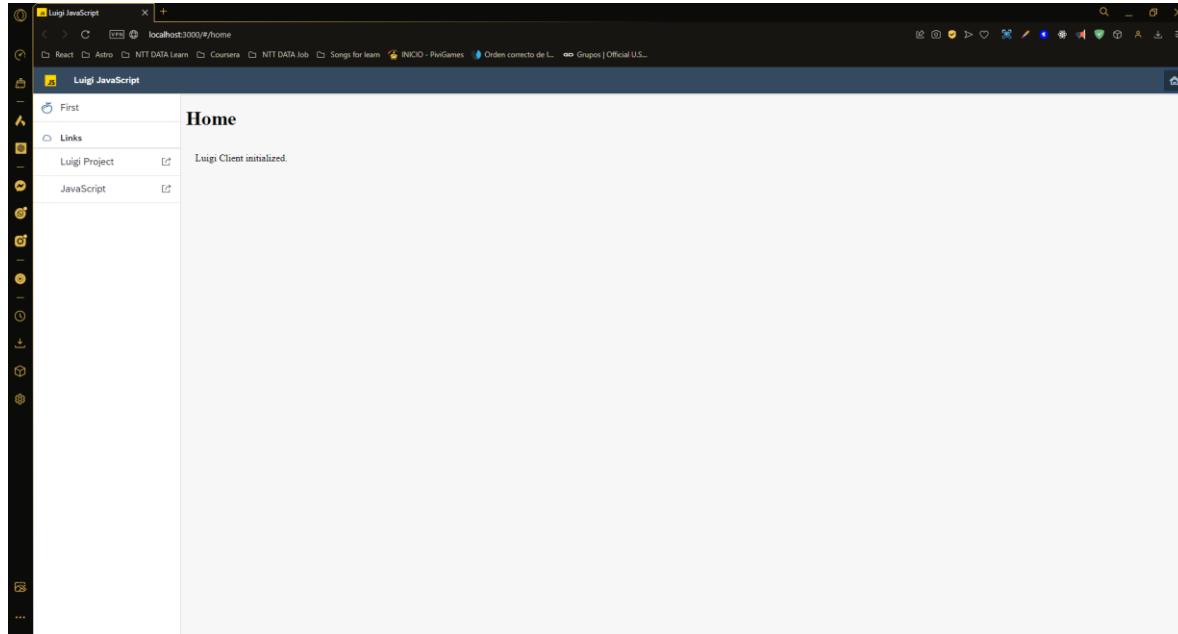
INFO Accepting connections at http://localhost:3000
```

Al finalizar la instalación, el proyecto se ejecutará automáticamente en el puerto 3000, vamos al navegador, y buscamos la ruta <http://localhost:3000>

SAP/Luigi: Microfrontend Framework



¡Ya tenemos la primera parte finalizada!, nuestro orquestador de microfrontends ya esta ejecutándose sin problemas 🤖



4. En la ubicación de nuestra carpeta luigi-microfrontend-app, abriremos un terminal y vamos a crear 3 microfrontends, para este ejemplo creare los 3 microfrontends en react, pero usted siéntase libre de escoger el framework/librería de su preferencia 🔥

El comando para crear nuestros microfrontends de react es:

```
npx create-react-app app1
```

Y le van cambiando el indicativo app1, app2, app3, etc...

SAP/Luigi: Microfrontend Framework



```
C:\Windows\System32\cmd.e  X  +  ▾

Microsoft Windows [Versión 10.0.22631.4391]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\david\OneDrive\Escritorio\luigi-microfrontend-app>npx create-react-app app1

Creating a new React app in C:\Users\david\OneDrive\Escritorio\luigi-microfrontend-app\app1.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1312 packages in 44s

259 packages are looking for funding
  run 'npm fund' for details

Initialized a git repository.

Installing template dependencies using npm...

added 46 packages, and changed 1 package in 6s

263 packages are looking for funding
  run 'npm fund' for details
Removing template package using npm...

removed 1 package, and audited 1358 packages in 4s

263 packages are looking for funding
  run 'npm fund' for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.

Created git commit.

Success! Created app1 at C:\Users\david\OneDrive\Escritorio\luigi-microfrontend-app\app1
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd app1
  npm start

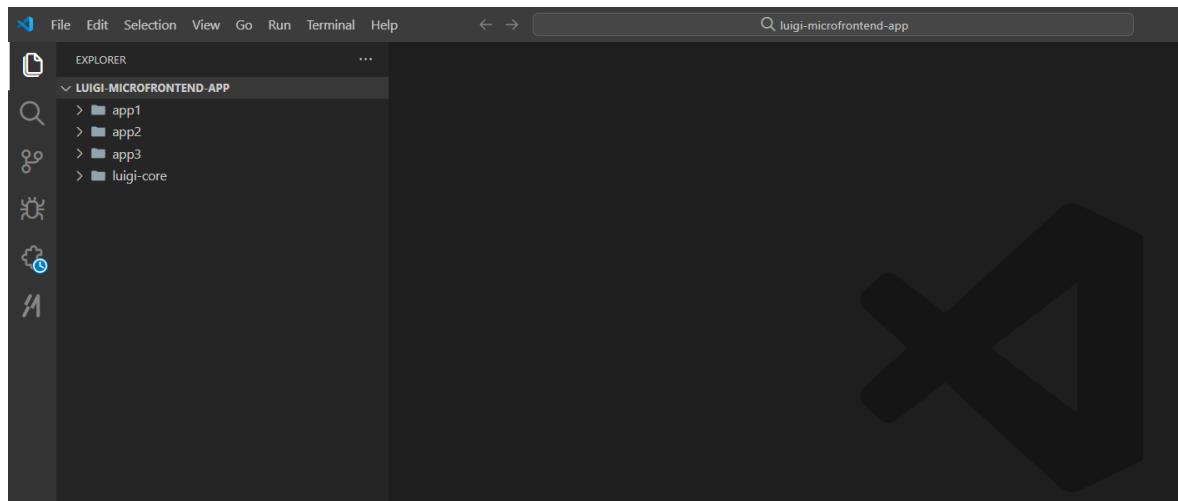
Happy hacking!

C:\Users\david\OneDrive\Escritorio\luigi-microfrontend-app>
```

SAP/Luigi: Microfrontend Framework

5. Una vez creados los 3 microfrontends, vamos a configurarlos para que corran en puertos diferentes.

Abriremos un Visual Code, desde la ubicación de nuestra carpeta principal luigi-microfrontend-app.



SAP/Luigi: Microfrontend Framework



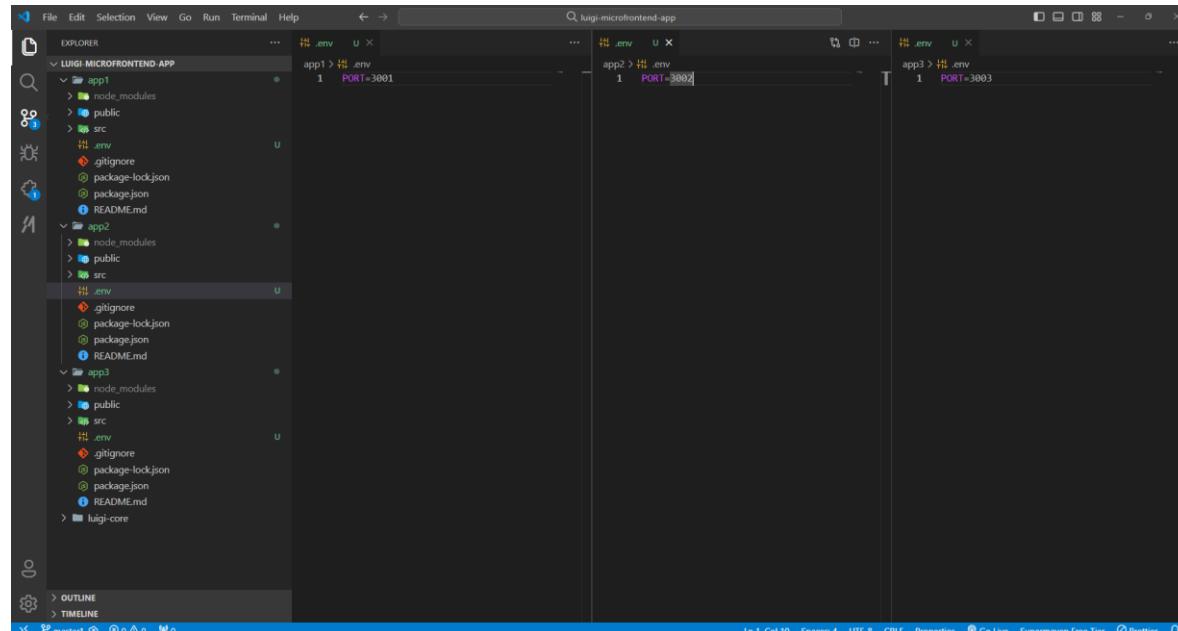
Vamos a crear un archivo llamado .env en cada carpeta de react: app1, app2 y app3

Una vez creado, agregaremos la siguiente línea en cada archivo respectivamente:

PORT=3001

PORT=3002

PORT=3003



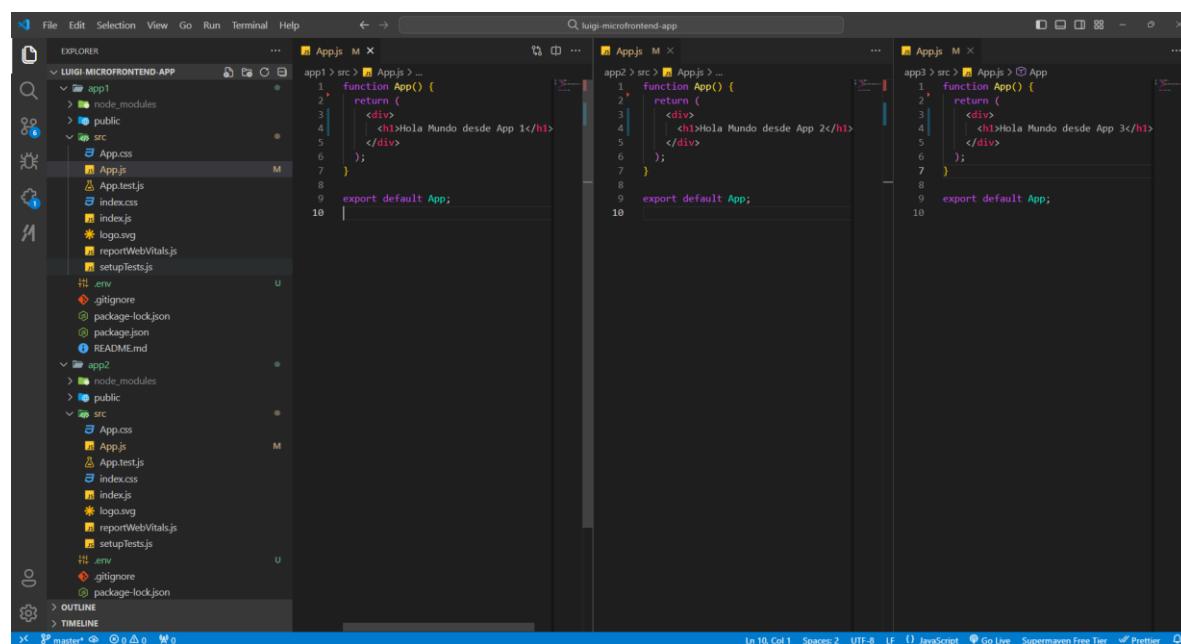
¡Debe quedar la configuración como se ilustra en la imagen!

SAP/Luigi: Microfrontend Framework

6. Modificaremos el archivo App.js de cada microfrontend, para agregarle un hola mundo desde app 1, 2 o 3.

Borraremos el contenido de esos archivos y lo reemplazaremos por el siguiente:

```
function App () {  
  return (  
    <div>  
      <h1>Hola Mundo desde App 1</h1>  
    </div>  
  );  
}  
  
export default App;
```



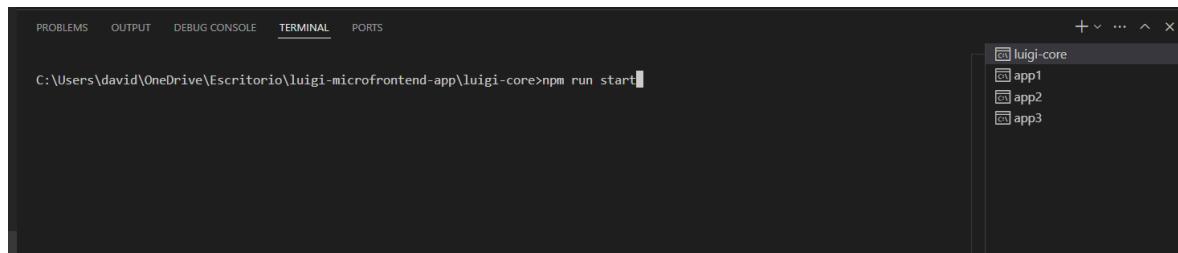
The screenshot shows a Visual Studio Code interface with three tabs open, each displaying the file 'App.js'. The tabs are labeled 'app1', 'app2', and 'app3'. The code in each tab is identical, showing a function named 'App' that returns a div containing an h1 element with the text 'Hola Mundo desde App [app number]'. The code is as follows:

```
function App () {  
  return (  
    <div>  
      <h1>Hola Mundo desde App 1</h1>  
    </div>  
  );  
}  
  
export default App;
```

7. Pondremos a correr todos los proyectos con el comando:

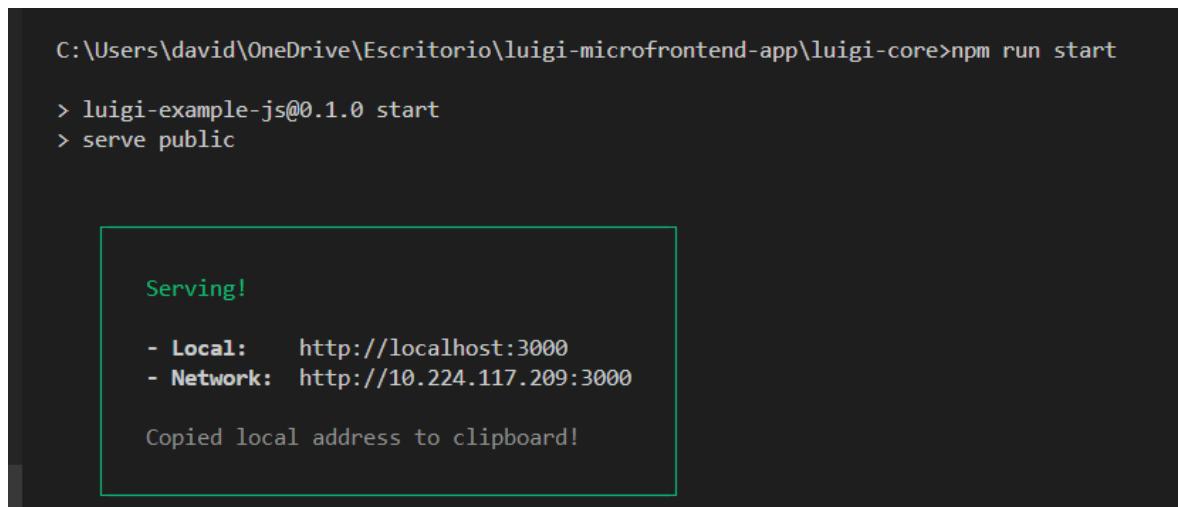
```
npm run start
```

Vamos a ejecutar todos los microfrontends y el luigi-core



A screenshot of a terminal window in a dark-themed IDE. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. In the terminal pane, the command `C:\Users\david\OneDrive\Escritorio\luigi-microfrontend-app\luigi-core>npm run start` is being typed. To the right of the terminal, there is a sidebar titled "PORTS" which lists four entries: luigi-core, app1, app2, and app3.

Esto nos debe indicar que los proyectos están ejecutándose en los puertos:
3000, 3001, 3002 y 3003



A screenshot of a terminal window showing the output of the `npm run start` command. The output shows two dependencies starting: `luigi-example-js@0.1.0 start` and `serve public`. A large green box highlights the following text:

Serving!

- **Local:** http://localhost:3000
- **Network:** http://10.224.117.209:3000

Copied local address to clipboard!

```
Compiled successfully!
```

```
You can now view app1 in the browser.
```

```
Local:          http://localhost:3001
On Your Network: http://10.224.117.209:3001
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
webpack compiled successfully
```

```
Compiled successfully!
```

```
You can now view app2 in the browser.
```

```
Local:          http://localhost:3002
On Your Network: http://10.224.117.209:3002
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
webpack compiled successfully
```

```
Compiled successfully!
```

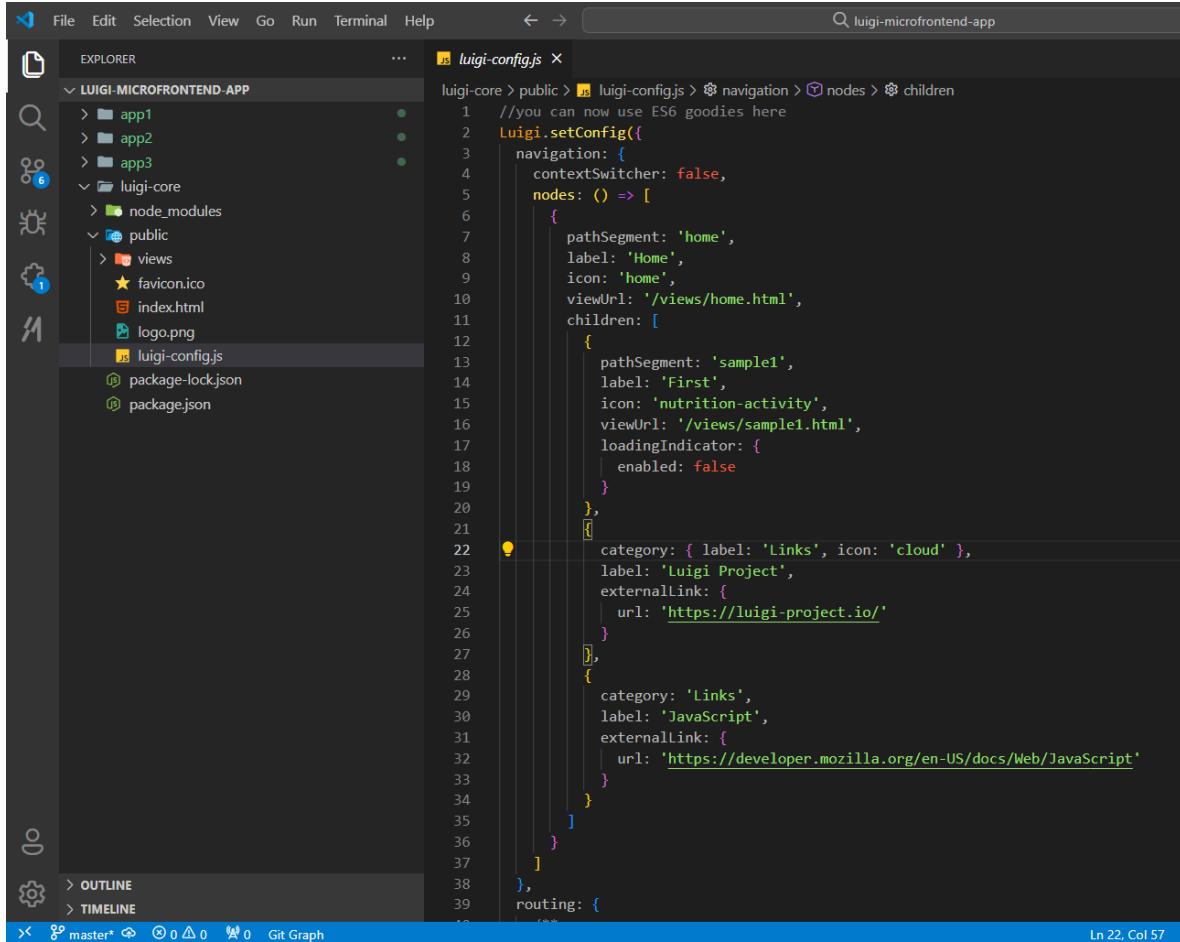
```
You can now view app3 in the browser.
```

```
Local:          http://localhost:3003
On Your Network: http://10.224.117.209:3003
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
webpack compiled successfully
```

8. Por último, vamos a configurar el archivo luigi-config.js que se encuentra en la carpeta public de luigi-core



```
//you can now use ES6 goodies here
Luigi.setConfig({
  navigation: {
    contextSwitcher: false,
    nodes: () => [
      {
        pathSegment: 'home',
        label: 'Home',
        icon: 'home',
        viewUrl: '/views/home.html',
        children: [
          {
            pathSegment: 'sample1',
            label: 'First',
            icon: 'nutrition-activity',
            viewUrl: '/views/sample1.html',
            loadingIndicator: {
              enabled: false
            }
          },
          {
            category: { label: 'Links', icon: 'cloud' },
            label: 'Luigi Project',
            externalLink: {
              url: 'https://luigi-project.io/'
            }
          },
          {
            category: 'Links',
            label: 'JavaScript',
            externalLink: {
              url: 'https://developer.mozilla.org/en-US/docs/Web/JavaScript'
            }
          }
        ]
      },
      routing: [
        ...
      ]
    }
  }
})
```

SAP/Luigi: Microfrontend Framework

Reemplazaremos su contenido con lo siguiente:

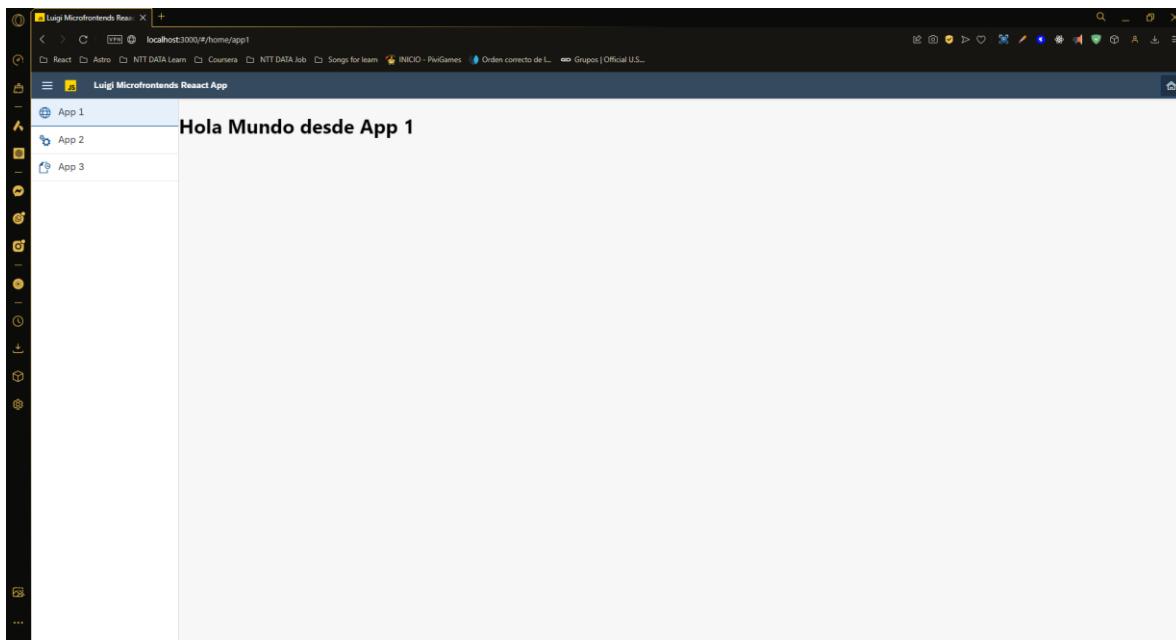
```
Luigi.setConfig({
  navigation: {
    contextSwitcher: false,
    nodes: () => [
      {
        pathSegment: "home",
        label: "Home",
        icon: "home",
        viewUrl: "/views/home.html",
        children: [
          {
            pathSegment: "app1",
            label: "App 1",
            icon: "world",
            viewUrl: "http://localhost:3001",
            loadingIndicator: {
              enabled: false,
            },
          },
          {
            pathSegment: "app2",
            label: "App 2",
            icon: "settings",
            viewUrl: "http://localhost:3002",
            loadingIndicator: {
              enabled: false,
            },
          },
        ],
      },
      {
        pathSegment: "app3",
        label: "App 3",
        icon: "timesheet",
        viewUrl: "http://localhost:3003",
        loadingIndicator: {
          enabled: false,
        },
      },
    ],
  },
  routing: {
    /**
     * Development:
     * For path routing, set to false
     * For hash routing, set to true
     */
    useHashRouting: true,
  },
  settings: {
    header: {
```

SAP/Luigi: Microfrontend Framework



```
title: "Luigi Microfrontends Reaact App",
logo: "/logo.png",
favicon: "/favicon.ico",
},
responsiveNavigation: "simple",
appLoadingIndicator: {
  hideAutomatically: true,
},
},
);
});
```

De esta manera, configuramos nuestro orquestador luigi, para que pueda mostrarnos los microfrontends de react que están corriendo en otros puertos, si siguieron todo el instructivo, finalmente tendrán una vista como la siguiente en su navegador:



Podrán navegar entre las opciones que configuramos previamente.

¡Muchas gracias por llegar hasta aquí! Espero esto les resulte útil para referencias y proyectos futuros 😊

Saludos,

Monsalve Delima, David Hernando | Engineer, Digital Architecture

dmonsalv@emeal.nttdata.com